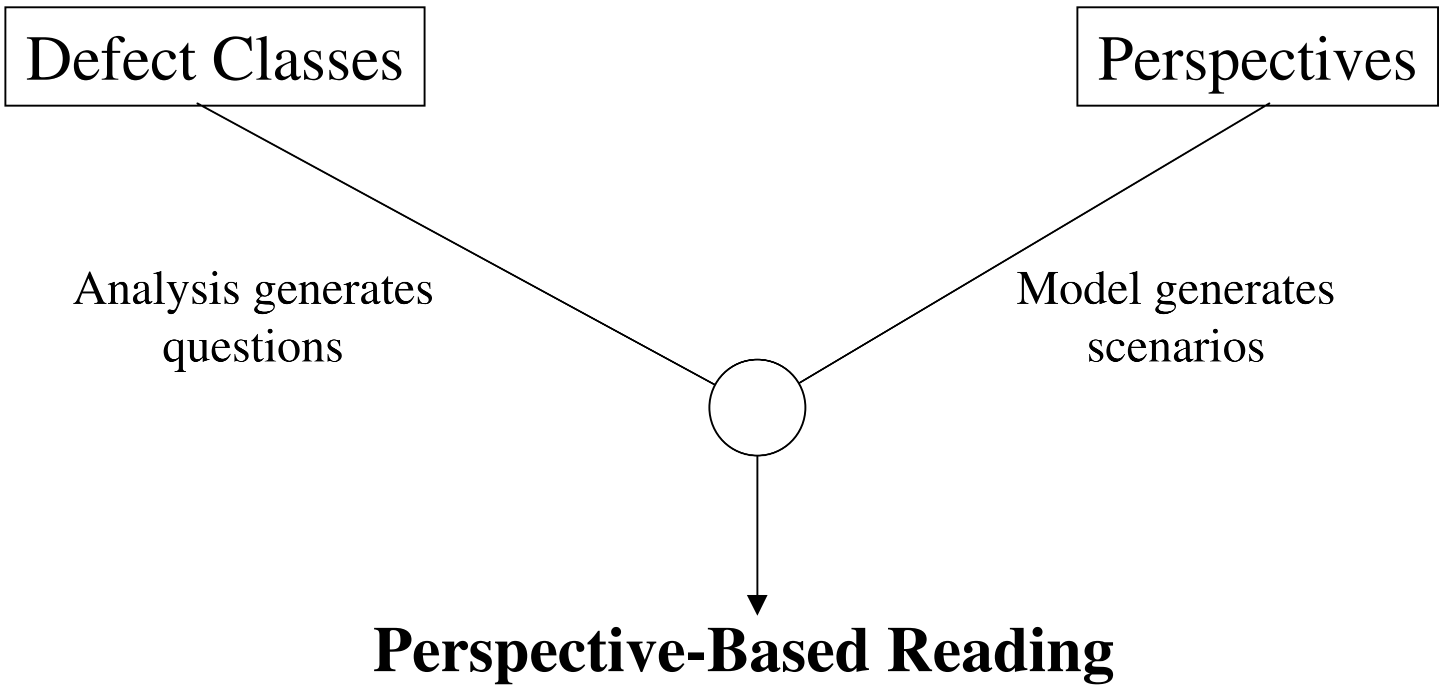
Perspective-Based Reading

- Example users of a requirements document:
  - a designer, who uses it to produce a system design;
  - a tester, who produces a test plan to ensure that the system meets the requirements;
  - a requirements analyst, who has to translate the functionality into a manual for the user;
  - …
- Each of them creates a model of the system described by the requirements.

Perspective-Based Reading



Perspective-Based Reading

- Goal for each participant:
  - **Analyze** the documents provided
  - **for the purpose of** evaluation
  - **with respect to** defect detection
  - **from the point of view of** designer (or tester, user, …)
  - **in the context of** a specific technique for creating system designs (or test plans, user manuals, …)

- Results:
  - defect report forms
  - a high-level model of the system

Test Perspective

- Focuses on a high-level form of **equivalence partition testing**
- For each requirement:
  - Read it through once and identify the inputs
  - For each input:
    - Divide the input domain into *equivalence sets*, where all of the values in each set will cause the system to behave similarly
    - For each equivalence set:
      - Identify test cases (select values which are typical as well as at the extremes of the equivalence set)
      - Record the system behavior which is expected to result when the test cases are input

Test Perspective

- **1. For each requirement, identify the inputs.**
  - Does the requirement make sense from what you know about the application of from what is specified in the general description?
  - Do you have all the information necessary to identify the inputs to the requirement? Based on the general requirements and your domain knowledge, are these inputs correct for this requirement?
  - Have any of the necessary inputs been omitted?
  - Are any inputs specified which are not needed for this requirement?
  - Is this requirement in the appropriate section of the document?

Test Perspective

- **2. For each input, construct equivalence sets.**
  - Do you have enough information to construct the equivalence sets for each input? Can you specify the boundaries of the sets at an appropriate level of detail?
  - According to the information in the requirements, are the sets constructed so that no value appears in more than one set?
  - Do the requirements state that a particular value should appear in more than one equivalence set (that is, do they specify more than one type of response for the same value)? Do the requirements specify that a value should appear in the wrong equivalence set?

Test Perspective

- **3. For each equivalence set, create test cases and specify the expected output.**
  - Do you have enough information to create test cases for each equivalence set?
  - Are there other interpretations of this requirement that the implementor might make based upon the description given? Will this affect the tests you generate?
  - Is there another requirement for which you would generate a similar test case but would get a contradictory result?
  - Can you be sure that the tests generated will yield the correct values in the correct units? Is the resulting behavior specified appropriately?

Example using Test Procedure

- **This example uses a requirements document which describes a system for running a video store. Here are some excerpts from the general description of this system:**
- "Customers select at least one video for rental. The maximal number of tapes that a customer can have outstanding on rental is 20. The customer's account number is entered to retrieve customer data and create an order. Each customer gets an ID card from ABC for identification purposes. This ID card has a bar code that can be read with the bar code reader. Bar code IDs for each tape are entered and video information from inventory is displayed. The video inventory file is updated. When all tape IDs are entered, the system computes the total bill. Money is collected and the amount is entered into the system. Change is computed and displayed. The rental transaction is created, printed and stored. The customer signs the rental form, takes the tape(s) and leaves. . . "

Example using Test Procedure

- Here is a brief description of the users of this system:
- “**User Characteristics:** The system will be used by ABC Video management, clerks, and indirectly by customers. From the point of view of the system, clerks and managers are identical. Some system operations are only accessible to managers (such as printing daily reports) and are protected by password. . .”

Example 1

- **Functional Requirement 8**
  - *Description*
    When all tapes are entered the system computes the total.
  - *Input*
    Enter key is pressed on keyboard after the last tape was entered.
  - *Processing*
    Computation of the total due. The total is the sum of the past-due fees, other fees and current video rental fees.
  - *Output*
    Total due.

Form A - Equivalence Sets and Test Cases for New Requirements

Reviewer Name: Document Reviewed:

Requirement Number: **8** Page:

Description of Input: **keyboard entry**

Valid Equiv. sets:	**Enter**		
Test cases:	**Enter**		
Test result:	**Compute Total**		
Invalid Equiv. sets:	**any other key**		
Test cases:	**Esc, space,...**		
Test result:			

What should happen when an invalid key is struck is not specified. However, is this omission going to lead to a problem with the implementation of the system? Probably not, so this should not be considered a fault in the requirements.

Description of Input:

Valid Equiv. sets:			
Test cases:			
Test result:			
Invalid Equiv. sets:			
Test cases:			
Test result:			

Example 2

- **Functional Requirement 7**
  - *Description*
    The maximum number of tapes that can be rented at one transaction is 20.
  - *Input*
    Bar code ID of tape is entered with the bar code reader
  - *Processing*
    If this tape is the 21st taken, rental is rejected
  - *Output*
    Error message is displayed

Form A - Equivalence Sets and Test Cases for New Requirements

Reviewer Name: Document Reviewed:

Requirement Number: **7** Page:

Description of Input: **# of tapes rented in this transaction**

Valid Equiv. sets:	**0-20**		
Test cases:	**0, 10, 20**		
Test result:	**wait for more input or end of transaction**		
Invalid Equiv. sets:	**<0**	**>20**	
Test cases:	**-1, -5, -maxint**	**21, 25, maxint**	
Test result:	**(not given)**	**reject rental, display error**	

Note that questions 1.1 and 1.2 may help uncover that there is a fault here, namely, that the requirement is testing the wrong thing (or alternately, that the wrong value is being input and checked to make sure that the customer has not exceeded their limit). (There is a fault because there is a discrepancy between Req. 7 and the test which is stated in the general description.)

Because what is in the general requirements is "truth", let us see if the rest of the requirement makes sense if we corrected the original fault by changing the input to "number of tapes currently rented."

Note that we make a distinction here between what happens if the customer tries to make too many rentals and what happens if the procedure gets a completely illegal value. In the latter case, the expected behavior is not specified in the requirements, so we have to make a judgement call as to whether or not this represents a fault: Do the requirements have to specify how the system recovers from an internal error, or can this be left to the discretion of the implementor?

Description of Input:

Valid Equiv. sets:			
Test cases:			
Test result:			
Invalid Equiv. sets:			
Test cases:			
Test result:			

Example 3

- **Functional Requirement 6**
  - *Description*
    Bar code IDs for each tape to be rented are entered.
  - *Input*
    Bar code IDs for each tape are entered with the bar code reader.
  - *Processing*
    Retrieving video inventory record about the tape.
  - O*utput*
    Display video name and price.

Form A - Equivalence Sets and Test Cases for New Requirements

Reviewer Name: Document Reviewed:

Requirement Number: **6** Page:

Description of Input: **tape ID**

Valid Equiv. sets:	**valid IDs**		
Test cases:	**(can't determine)**		
Test result:			
Invalid Equiv. sets:	**invalid (illegal, unassigned) IDs**		
Test cases:	**(can't determine)**		
Test result:			

Description of Input:

Valid Equiv. sets:			
Test cases:			
Test result:			
Invalid Equiv. sets:			
Test cases:			
Test result:			

Notice that the requirement does not give any information on what a valid or invalid ID looks like. If the information is omitted from the rest of the document as well, this is something that should be noticed as soon as we try to construct the test cases. Again, whether or not this omission is an actual fault depends on a judgement call as to whether or not this information should be the prerogative of the implementor.